INRIA
CENTRE DE ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

N° 545

CLASSIFICATION AUTOMATIQUE DES DONNÉES TECHNIQUES DE PRODUCTION

Hervé GARCIA
Bernard MUTEL
Jean-Marie PROTH

Juillet 1986

CLASSIFICATION AUTOMATIQUE

DES DONNEES TECHNIQUES DE PRODUCTION

Hervé GARCIA *
Bernard MUTEL *
Jean-Marie PROTH **

* : Laboratoire d'Automatique et d'Electronique Industrielles, Université de Metz, IUT, Ile du saulcy, 57045 METZ CEDEX

** : INRIA-LORRAINE, Rue du Doyen Roubault, Château du Montet, 54500 VANDOEUVRE

Ce travail a été financé par l'Agence de l'Informatique (ADI).

PLAN

I - INTRODUCTION

II - CODAGE DES DONNEES DE PRODUCTION

- II-1- Choix d'un code
- II-2- Représentation mathématique du code
- II-3- Exemple
- II-4- Cas des gammes de fabrication

III - RECHERCHE DE FAMILLES DE PIECES

- III-1- Mesure de ressemblance
- III-2- Cas de variables nominales
- III-3- Cas de variables ordinales
- III-4- Distance entre individus
- III-5- Algorithme de classification
  - III-5-1- Critères
  - III-5-2- Algorithme
  - III-5-3- Propriétés
  - III-5-4- Initialisation
  - III-5-5- Validation des classes
- III-6- Exemple

IV - Classement d'une nouvelle pièce

- IV-1- Heuristique de Classement
- IV-2- Remplacement
- IV-3- Cas de distances sur plusieurs groupes

V - CONCLUSION

BIBLIOGRAPHIE

ANNEXE

ABSTRACT

This paper is devoted to the coding in production. It gives a new algorithm for finding part families. It also proposes an affectation algorithm.

RESUME

Dans ce papier, nous examinons les problèmes posés par la codification en production. On y donne un algorithme nouveau de constitution de familles de produits. On y donne également un algorithme d'affectation.

I - INTRODUCTION

La nécessité, pour les entreprises, d'améliorer leur productivité implique une intégration des fonctions du système complet de production. Cette intégration se caractérise en particulier par des échanges accrus d'informations pertinentes entre tous les services de l'entreprise.

Le volume des données commerciales et techniques à prendre en compte exige la mise en oeuvre de processus capables de les filtrer et de les regrouper dans le but d'optimiser leur flux et de faciliter la préparation des solutions issues de l'expérience.

C'est l'un des aspects de la Technologie de Groupe : on cherche à regrouper les produits en familles ou groupes présentant des caractéristiques similaires du point de vue de leur conception et/ou de leur fabrication et/ou de leur assemblage, etc.

Dans la plupart des logiciels de la Technologie de Groupe actuellement sur le marché, la recherche des familles s'effectue simplement par des procédures de tri selon les valeurs des modalités des variables du code représentant les données de production, c'est-à-dire les données qui décrivent la pièce et les indications qui permettent de la produire.

Cette technique de "filtrage par peigne" est insuffisante selon nous pour deux raisons principales :

- la première raison a trait à l'utilisation des codes. Dans le "filtrage par peigne", le tri s'effectue à partir de valeurs fournies par l'utilisateur. Ces valeurs initiales sont déterminantes pour le résultat, qui s'en trouve ainsi faussé.

- la seconde raison a trait à la technique. Le "filtrage par peigne" ne permet en effet que de regrouper des produits très voisins au sens des valeurs des paramètres retenus pour effectuer la sélection, ce qui est très limitatif. En effet, avec cette méthode, deux produits ne figureront dans la même classe que si **tous** les paramètres retenus par l'utilisateur pour effectuer le traitement ont des valeurs voisines. Or, nous savons bien que deux produits peuvent être "proches" même si, par exemple, les valeurs d'un des paramètres diffèrent beaucoup, à condition, bien entendu, que les valeurs des autres paramètres soient très voisines. La Technologie de Groupe doit, nous semble-t-il, être en mesure d'intégrer ces aspects, surtout dans un monde où l'économie d'envergure remplace progressivement l'économie d'échelle.

C'est ainsi que, depuis plusieurs années, des travaux ont été entrepris pour appliquer les méthodes d'analyse des données, et en particulier de classification automatique, aux problèmes de reconnaissance de familles de produits (1.,2.,3.,4.,5.)

Dans ce qui suit, nous analysons les codes de représentation de pièces mécaniques utilisés dans l'industrie, puis nous proposons une méthode de classification automatique permettant de constituer des familles de pièces.

II - CODAGE DES DONNEES DE PRODUCTION

II-1- CHOIX D'UN CODE

Depuis fort longtemps, on a cherché a identifier les paramètres caractéristiques des produits à fabriquer ainsi que les moyens de fabrication et de transport. Ceci a donné lieu à la définition de nombreux codes de représentation des produits. La figure 1 recense les caractéristiques des codes de représentation de pièces mécaniques les plus utilisés.

D'une façon générale, les paramètres caractéristiques d'un produit dépendent des objectifs de classification à réaliser. La figure 2 donne quelques uns de ces objectifs pour les services techniques.
A chacun de ces objectifs va correspondre un code particulier. Ce code général C de représentation d'un produit sera l'union des codes C_i pour chaque objectif f_i :

$$C = \bigcup_i C_i$$

Afin d'éviter la redondance d'informations, la réunion des codes se fait avec le "ou exclusif" noté V :

$$C = \bigvee_i C_i$$

Le nombre de grandeurs caractéristiques d'un code (variables) ne doit être ni trop faible ni trop grand.

		CODES	ACAPS	SAGT	CETIM	TEKLA	TNO	OPITZ	GIR
		Nombre de positions	21	18	6	12	12	13	19
CARACTERISTIQUES	MORPHOLOGIE	- Pièces de révolution							
		. sans dimension			x				
		. dimensions	x	x		x	x	x	x
		. rapport L/D							
		- Pièces de non révolution							
		. sans dimension			x				
		. dimensions	x			x	x	x	x
		. rapport de dimensions	x				x	x	x
		- Formes							
		. extérieures	x	x	x	x	x	x	x
		. intérieures	x	x	x	x	x	x	x
		. complémentaires	x	x	x	x	x	x	x
		. complexité		x				x	x
	FONCTION	Type de Pièces	x						x
	FABRICATION	Matière première	x	x	x	x	x	x	x
		Forme initiale			x		x	x	x
		Tolérances	x	x		x	x	x	x
		Etat de surface		x		x	x		x
		Quantité		x			x	x	x
		Poids						x	
		Axes d'usinage							x
		Surfaces non usinées	x						
		Temps de fabrication	x						

Code de représentation de pièces mécaniques

Fig. 1

Bureau d'étude

- standardisation des formes
- classification

Bureau des méthodes

- normalisation des procédés de fabrication
- détermination des gammes types
- détermination des temps regroupés
- normalisation de l'outilllage

Atelier

- organisation en îlots
- transport
- assemblage

Domaines d'application de la T.G.

Fig. 2

Un faible nombre de paramètres facilite l'écriture et l'interpétation du code. Cependant, si ce nombre est trop restreint, la représentation est imprécise et ne permet pas de faire apparaître des familles suffisamment homogènes.

Par contre, un nombre important de paramètres revient à noyer les paramètres caractéristiques les plus pertinents dans un amas de caractéristiques secondaires. Ceci introduit un "bruit de fond" important qui influence défavorablement les résultats. On sait, en effet, qu'une éventuelle pondération des variables est, en pratique, difficile et conduit à des résultats peu significatifs. La pondération est d'ailleurs impossible dans les "filtrages par peigne".

REMARQUE

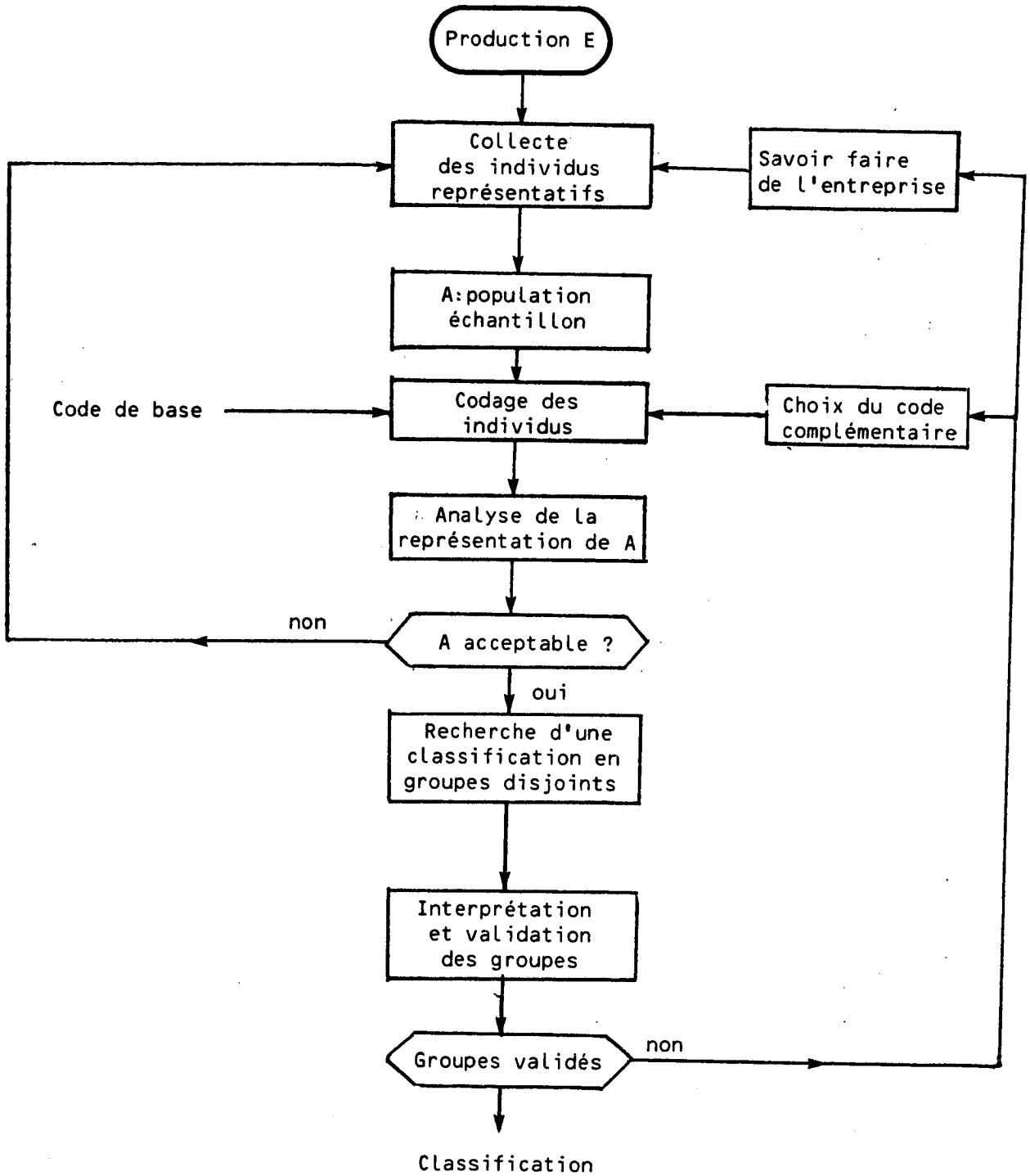
Le problème des codes en FAO (Fabrication Assistée par Ordinateur) que nous traitons ici restera longtemps d'actualité, malgré les tentatives intéressantes pour le déduire automatiquement des résultats de la CAO.

Soulignons d'ailleurs que le code FAO s'obtient en interprétant le code CAO à la lumière de connaissances technologiques. Ceci rend le passage automatique de CAO et FAO peu probable.

En pratique, les codes industriels comportent de 5 à 50 variables. On choisit un code de base C_b existant, se rapprochant le plus de l'objectif à atteindre et du type de production concernée. A ce code de base, on adjoint un code complémentaire C_c spécifique de l'application envisagée et des particularités de l'entreprise. Le choix de ce code complémentaire est délicat lors d'applications nouvelles.

Le code optimal résulte de la classification. Il s'obtient en dégageant les paramètres caractéristiques des classes obtenues. Il s'agit d'une analyse discriminante. Il faut cependant souligner que le code dit "optimal" reste dans l'espace des paramètres initialement retenus. Si donc le code initial ne contient pas explicitement ou implicitement les paramètres caractéristiques que l'on recherche, il ne faut espérer aboutir à un code "optimal" ou simplement acceptable. En outre, la classification devrait théoriquement se faire à partir de l'ensemble de la population. Mais cela exigerait la connaissance du code complet de tous les éléments de cette population. Or l'établissement du code complémentaire C_c est long est difficile. On se contente donc généralement de traiter un échantillon de population jugé significatif.

La figure 3 résume le processus de choix du code.



Processus de choix du code de production

Fig. 3

La population échantillon A est un sous-ensemble de la population totale E. A est souvent choisi de telle sorte qu'elle représente au mieux la majorité des produits de E. Ceci élimine les sous-populations peu nombreuses.

II-2- REPRESENTATION MATHEMATIQUE DU CODE

Le choix d'une population échantillon et d'un code étant effectué, nous sommes en présence d'une collection de paramètres (ou variables) caractéristiques . La majorité des codes ne considère que des variables nominales comportant chacune 10 modalités. Il y a ainsi perte d'information (Dégénérescence des variables métriques et ordinales en variables nominales).

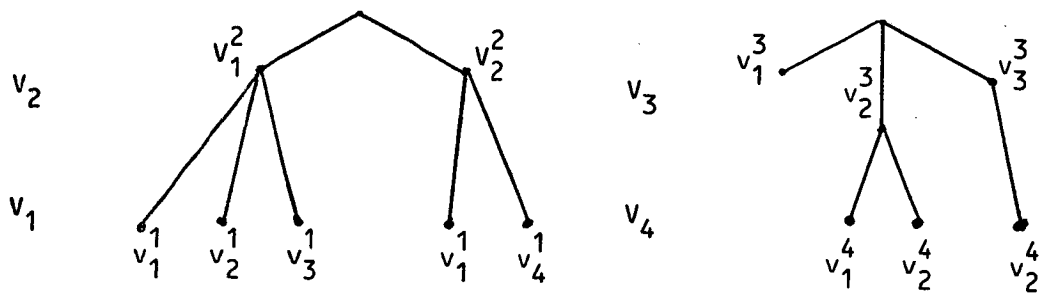
Soit un produit x (individu) appartenant à A. x est représenté par p variables v^j, et card A = n

$$x = (v^1, v^2, \ldots, v^p)$$

Afin d'avoir une représentation visuelle approchée de la répartition des produits dans R^p et de la répartition des modalités dans R^n il est souvent intéressant d'effectuer une analyse fonctionnelle des correspondances (3.). L'analyse des premiers plan factoriels permet de se rendre compte rapidement de la pertinence du code et des modalités ainsi que de la distribution des produits dans A.

Dans l'industrie, chaque code se représente sous forme d'une ou plusieurs arborescences. Deux variables dépendantes se trouvent dans la même arborescence. Deux variables indépendantes se trouvent dans des arborescences différentes. C'est au moins l'objectif. La figure 4 schématise cette situation pour l'exemple donné par le tableau suivant :

variables / valeurs	V_1	V_2	V_3	V_4
	v_1^1	v_1^2	v_1^3	v_1^4
	v_2^1	v_2^2	v_2^3	v_2^4
	v_3^1		v_3^3	
	v_4^1			



Structure d'un code

Fig. 4

Dans ce cas de figure, V_1 et V_2 sont dépendantes de même que V_3 et V_4. On remarquera que le graphe et le tableau ne sont pas en correspondance biunivoque. Par contre un élément quelconque du couple (V_1,V_2) est indépendant d'un élément quelconque du couple (V_3,V_4).

Ces codes se prêtent mal à l'analyse des données qui demande, pour être objective, un poids identique pour chaque variable. L'indépendance des variables est un facteur supplémentaire d'efficacité des techniques d'A.D. Ainsi, en classification automatique, on est amené à effectuer un transcodage T sur les individus.

$$x = (x^1, v^2, \ldots, v^p) \xrightarrow{T} (x^1, x^2, \ldots, x^q)$$

L'algorithme de transcodage dépend du code. Il a pour but de restituer des variables indépendantes auxquelles on attribuera le même poids. L'ensemble des individus constitue le tableau de données X à classifier, où x_i^j est la valeur prise par la $j^{ème}$ variable de l'individu x . Afin qu'il n'y ait pas de perte d'information lors du transcodage, nous construisons en pratique deux tableaux de n individus. Le premier comporte en colonne les q_1 variables nominales et l'autre les q_2 variables ordinales. $(q_1+q_2 = q)$

II-3- EXEMPLE : CODE CETIM OTP (fig. 5)

Le code de base comporte trois caractéristiques (rangs).

La caractéristique de rang 3, qui indique le rapport longueur sur diamètre pièce, est une variable ordinale.

L'interprétation des modalités de la caractéristique de rang 2 dépend étroitement de la valeur des modalités du rang précédent.

Ainsi, pour $v^1 = (0,1,2,3,4)$ la signification de v^2 est donnée par le tableau.

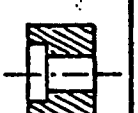
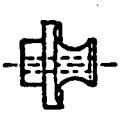
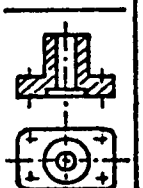
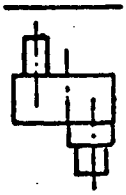
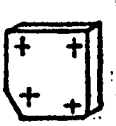
Pour $v^1 = 5$ ou 6 ou 7 ou 8 ou 9, les modalités de la variable v^2 auront une autre singification.

GROUPEMENT ANALOGIQUE DES PIECES

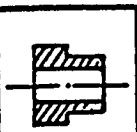
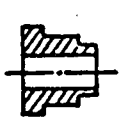
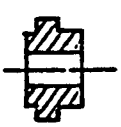
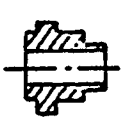
CODE CETIM O.T.P.

CODE PRIMAIRE MORPHODIMENSIONNEL

Rang 1 - TYPE DE PIECE

DE REVOLUTION					COMPOSE AVEC FORME DE REVOLUTION		NON DE REVOLUTION		
PLEINE	AVEC TROU BORGNE	AVEC TROU DEBOUCHANT		AUTRE	SUIVANT 1 AXE	MULTI-AXE	PARALLELEPIPEDIQUE		AUTRE
		NON ETAGE	ETAGE				PLATE		
									
0	1	2	3	4	5	6	7	8	9

Rang 2

			—			—	—		—
0	1	2	3	4	5	6	7	8	9

Rang 3

L/D ≤ 1/10	> 1/10 ≤ 1/5	> 1/5 ≤ 1/3	> 1/3 ≤ 1/2	> 1/2 ≤ 3/4	> 3/4 ≤ 1	> 1 ≤ 1,5	> 1,5 ≤ 2	> 2 ≤ 3	L/D > 3
0	1	2	3	4	5	6	7	8	9

Fig. 5

II-4- CAS DES GAMMES DE FABRICATION

Au paragraphe II-2 nous avons discuté du choix des variables d'un code lorsque celles-ci ne simposaient pas. C'est le cas par exemple de la représentation d'un plan de détail, d'un devis,.... Ce choix des variables pour représenter les gammes de fabrication est évident : c'est la suite des opérations de la gamme. Par contre, le choix d'une représentation mathématique est délicat car nous sommes en présence d'une suite ordonnée incomplète de modalités. Nous avons proposé deux méthodes pour représenter ce type de problèmes : l'une est basée sur un transcodage pour se ramener à une matrice logique (11.), l'autre est basée sur une méthode de reconnaissance structurelle de gammes utilisant la programmation dynamique (12.). Cette dernière utilise comme critère le coût d'apparaiement entre gammes en tenant compte de contraintes techniques (machine multi-opérations, gammes de remplacement,...) et de contraintes économiques (charges des machines,...).

III - RECHERCHE DE FAMILLES DE PIECES

III-1- DEFINITION D'UNE MESURE DE RESSEMBLANCE ENTRE INDIVIDUS

Soit E l'ensemble des individus. On définit une application d de E x E dans R^+. d est une distance si elle respecte les axiomes suivants :

- $d(x_i,x_j) = 0$ si $i = j \quad \forall \; i,j \in E$
- $d(x_i,x_j) = d(x_j,x_i)$
- $d(x_i,x_j) \leqslant d(x_i,x_k) + d(x_k,x_j) \quad \forall \; k \in E$

Nous distinguons respectivement deux distances sur les tableaux X_1 et X_2 à q_1 et q_2 variables.

III-2- CAS DES VARIABLES NOMINALES

Nous construisons un tableau disjonctif complet sur les q_1 variables nominales pour obtenir un tableau logique T_1. A partir de ce tableau nous définissons la distance entre individus x_i et x_j selon la distance de Hamming pondérée :

$$d_1(x_i,x_j) = \sum_{r=1}^{q_1} \alpha_1^r (x_i^r - x_j^r)^2$$

α_1^r pondération sur les variables.

D'autres mesures de distance peuvent être choisies à partir d'indices de ressemblance définis sur l'ensemble des variables logiques. Il faut cependant s'assurer que les dissimilarités correspondantes présentent bien les propriétés des distances Euclidiennes (7.).

III-3- CAS DE VARIABLES ORDINALES

Nous considérons indifféremment les variables numériques et ordinales. La distance entre deux individus est :

$$d_2(x_i, x_j) = \sum_{r=q_1+1}^{q} \alpha_2^r (x_i'^r - x_j'^r)^2$$

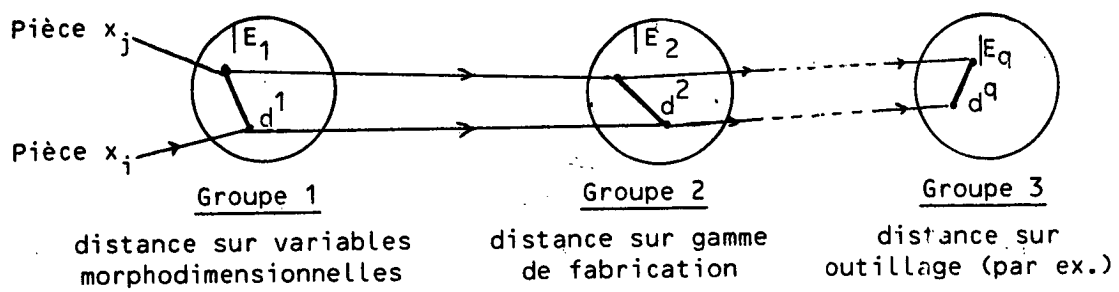
$$x'^r = \frac{x^r - \bar{x}^r}{\sigma_r} \qquad \bar{x}^r = \frac{1}{n} \sum_{i=1}^{n} x_i^r$$

$$\sigma_r^2 = \frac{1}{n} \sum_{i=1}^{n} (x_r^r - \bar{x}_i^r)^2$$

III-4- DISTANCE GLOBALE ENTRE INDIVIDUS

Nous venons de calculer les distances entre pièces représentées par des variables du groupe morphodimensionnel. Mais ces mêmes pièces peuvent être également représentées par des variables d'autres groupes comme indiqué en II. 4 pour les gammes de fabrication.

Ainsi, dans le cas général, l'apparaiement entre deux individus (pièces) se fait à partir d'une distance globale, somme des distances sur différents types de variables des différents groupes (fig. 6). Cette distance globale est un scalaire et l'on n'a plus la notion de nuage de points dans R^p.



Distance globale entre individus

Fig. 6

Pour différents groupes $d(x_i, x_j) = \sum_{r=1}^{q} \alpha_r \, d_{ij}^r$

α_r pondération entre les groupes.

Dans ce qui suit nous ne considérons qu'un seul groupe E_1.

La distance entre deux individus est :

$$d(x_i, x_j) = \frac{\lambda 1}{h_1} \, d_1(x_i, x_j) + \frac{\lambda 2}{h_2} \, d_2(x_r, x_j)$$

- λ_1, λ_2 : pondération choisie par l'utilisateur entre les tableaux des variables qualitatives et quantitatives. Normalement : $\lambda_1 = \frac{1}{q_1}$ et $\lambda_2 = \frac{1}{q_2}$, où q_1 et q_2 sont les tailles des groupes.
- h_1, h_2 : pondération sur les distances afin que leur contributions aient le même poids. Généralement h_1=1 et h_2=1, sinon h_1 est la plus grande valeur propre de la matrice de variance covariance du tableau T, de même pour h_2 (8.)

III-5- ALGORITHME DE CLASSIFICATION

III-5-1- CRITERE

Soit T le tableau symétrique croisant les distances entre pièces.

On recherche une partition P de T en K classes disjointes telle que toute pièce x_i appartenant à une classe P_l soit plus proche des autres pièces $x_j \in P_l$ que des pièces $x_r \notin P_l$

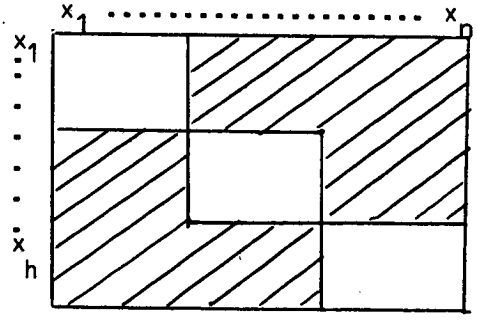


Tableau T

$$P = (P_1, \ldots, P_K)$$

soit $x_i \in P_l$

$$d(x_i, x_j) \leq d(x_i, x_r) \quad \left.\begin{array}{l} \forall\, i,j \in P_l \\ \forall\, r \notin P_l \end{array}\right\} \; l = 1,2,\ldots,K$$

Ceci vient à rechercher K blocs diagonaux tels que les distances à l'intérieur de ces blocs soient inférieures à n'importe quelle distance hors bloc.

En pratique, nous affectons un poids m_i à chaque individus x_i afin de tenir compte de l'importance de la série de pièces, de la plus value, etc.

$$\text{Soit } W(P) = \sum_{p=1}^{K} \frac{1}{2M_l} \sum_{j \in P_l} \sum_{i \in P_l} m_i \, d(x_i, x_j)$$

Une bonne solution du problème précédent est la recherche d'une partition P* qui minimise (W(P)).

$$W(P^*) = \min_{P} W(P)$$

III-5-2- ALGORITHME

1. Initialisation. Partition initiale, $W^0 = \infty$, nombre de classes K

2. Pour chaque individu x_i, i=1,...,n
   2.1. Pour chaque classe P_l, l=1,...,K
      2.1.1. Calculer

$$D(x_i, P_l) = \frac{1}{M_l} \sum_{j \in P_l} m_j \, d(x_j, x_i)$$

$$M_l = \sum_{j \in P_l} m_j$$

3. Pour chaque individu x_i, i=1,...,n
   3.1. Affecter x_i à P_{l^*} tel que

$$D(x_i, P_{l^*}) = \min_{l} D(x_i, P_l)$$

4. Calculer $W' = \frac{1}{2} \sum_{l=1}^{K} \frac{1}{M_l} \sum_{\substack{j \in l \\ i \in l}} (m_i \, d(x_i, x_j))$

avec $M_l = \sum_{j \in l} m_j$

5. Calculer $W^0 - W' = \delta$
   5.1. Si $\delta > \varepsilon$ donné, alors $W^0 = W'$ et retourner en 2,
   sinon, aller en 6.

6. Fin de procédure

III-5-3- PROPRIETES DE L'ALGORITHME

THEOREME : Si q est l'indice d'itération de l'algorithme alors la suite des valeurs prises par W^0 est une suite décroissante convergente.

DEMONSTRATION : Nous avons $W^0 \geq 0$ à chaque itération.

A la $q^{ème}$ itération

$$v_q = W^0 = \sum_{l=1}^{K} \sum_{i \in P_l} D(x_i, P_{l*}^{(q)})$$

A l'itération (q+1), $P^{(q+1)} = (P_1^{(q+1)}, \ldots, P_K^{(q+1)})$ est construit tel que $D(x_i, P_{l*}^{(q+1)}) = \underset{l}{\text{Min}}\ D(x_i, P_l^q)$,

d'où $v_{q+1} \leq v_q$.

La suite v_q est donc décroissante et converge en un nombre fini d'itérations car card (T) est fini.

Cet algorithme de classification peut être considéré comme une extension de la méthode décrite en (10.) et comme une variante de la méthode des nuées dynamiques (9.) (noyau partition avec masses variables). Le tableau des distances étant donné une fois pour toutes, l'algorithme ne nécessite à chaque itération que le calcul des sommes des distances intra et inter classes.

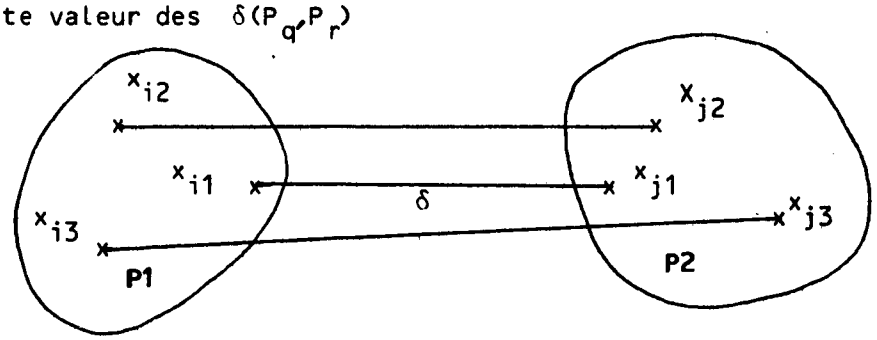
III-5-4- INITIALISATION

Compte tenu du fait que nous diposons des distances entre individus, l'initialisation se fait par une classification hiérarchique ascendante avec aggrégation des classes par diamètre maximum.

Les classes P_1 et P_2 seront réunies si

$$\delta(P_1, P_2) = \text{Max } d(x_i, x_j) \quad \forall\, x_i \in P_1,\ x_j \in P_2$$

est la plus petite valeur des $\delta(P_q, P_r)$



Du point de vue pratique, cette initialisation permet de se fixer un nombre de classes K compte tenu de la dispersion maximale entre individus que l'on se donne sur les classes. Ceci permet éventuellement d'éliminer des individus trop disparates par rapport à la population.

REMARQUE : Le tableau T étant symétrique et de diagonale nulle, du point de vue informatique on ne mémorise que $\frac{n(n-1)}{2}$ distances.

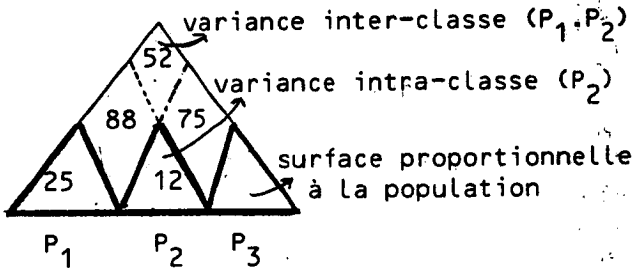
III-5-5- VALIDATION DES CLASSES

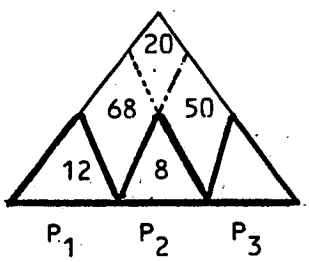
Quelle que soit la méthode de classification mise en oeuvre, seul l'utilisateur peut valider formellement les classes obtenues. En particulier, il faut tester la stabilité des classes obtenues vis à vis de différentes partitions initiales et vis à vis d'une modification de l'échantillon de la population. En (13.) sont exposés différents tests de validation utilisables par l'analyste.

Plusieurs types de représentations graphiques sont possibles pour visualiser les résultats obtenus.

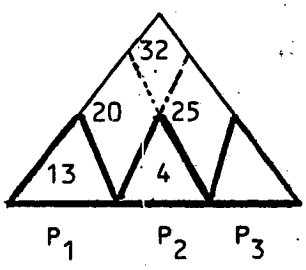
La représentation graphique que nous donnons est expliquée par l'exemple suivant :

EXEMPLE : 3 classes P_1, P_2, P_3 et 2 groupes de variables :





Contribution du
Premier groupe de variables
qualitatives



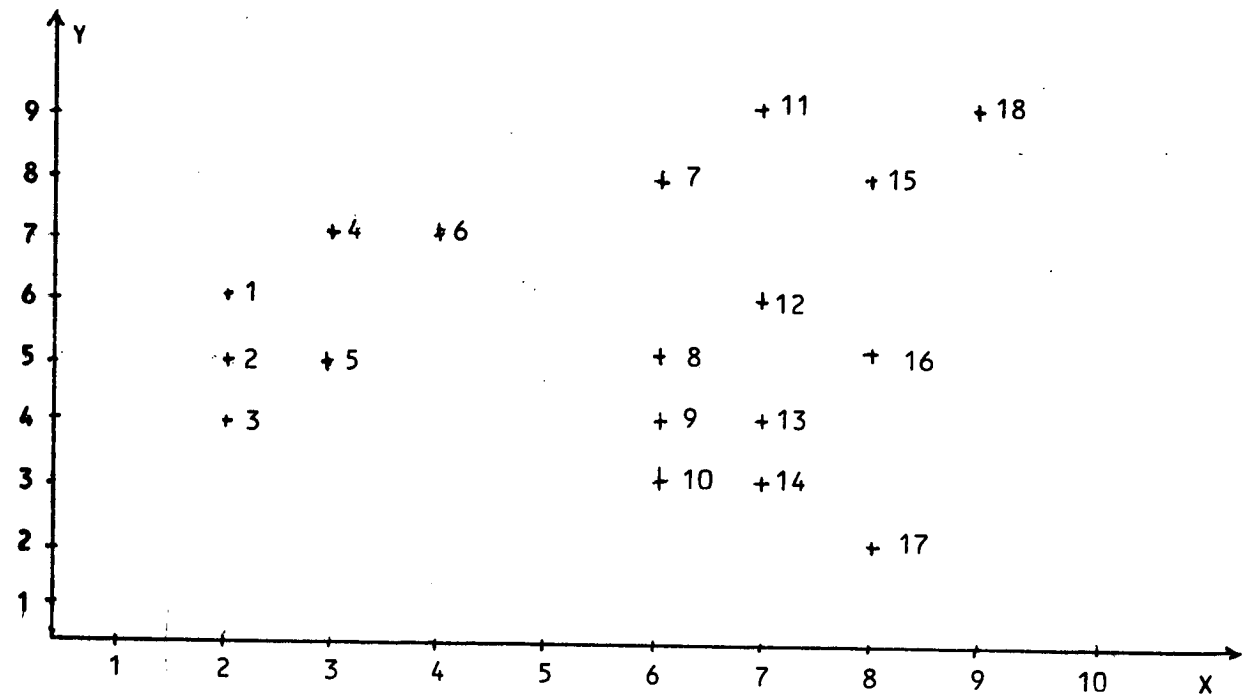
Contribution du
Deuxième groupe de variables
qualitatives

III-6- EXEMPLES

III-6-1- EXEMPLE 1

Il s'agit de tester l'algorithme à partir de données représentant 18 individus définis par deux paramètres (fig. 7.1). Les distances correspondantes sont données par le tableau 7.2 qui fournit également la partition initiale pour 3 classes. Au bout de trois itérations, l'algorithme converge vers les résultats suivants :

- classe 1 : 6,4,5,3,2,1
- classe 2 : 11,7,18,15
- classe 3 : 8,9,10,12,13,14,17,16



	X,Y		X,Y		X,Y		X,Y
1	2,6	6	4,7	11	7,9	16	8,5
2	2,5	7	6,8	12	7,6	17	8,2
3	2,4	8	6,5	13	7,4	18	9,9
4	3,7	9	6,4	14	7,3		
5	3,5	10	6,3	15	8,8		

Distribution des données

Fig. 7.1

Matrice de distances - partition initiale

		1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	1	0	1	4	2	2	5	20	17	20	25	34	25	29	34	40	37	52	58
2	1	1	0	1	5	1	8	25	16	17	20	41	26	26	29	45	36	45	65
3	1	4	1	0	10	2	13	32	17	16	17	50	29	25	26	52	37	40	74
4	1	2	5	10	0	4	1	10	13	18	25	20	17	25	32	26	29	50	40
5	1	2	1	2	4	0	5	18	9	10	13	32	17	17	20	34	25	34	52
6	1	5	8	13	1	5	0	5	8	13	20	13	10	18	25	17	20	41	29
7	2	20	25	32	10	18	5	0	9	16	25	2	5	17	26	4	13	40	10
8	2	17	16	17	13	9	8	9	0	1	4	17	2	2	5	13	4	13	25
9	2	20	17	16	18	10	13	16	1	0	1	26	5	1	2	20	5	8	34
10	2	25	20	17	25	13	20	25	4	1	0	37	10	2	1	29	8	5	45
11	2	34	41	50	20	32	13	2	17	26	37	0	9	25	36	2	17	50	4
12	2	25	26	29	17	17	10	5	2	5	10	9	0	4	9	5	2	17	13
13	3	29	26	25	25	17	18	17	2	1	2	25	4	0	1	17	2	5	29
14	3	34	29	26	32	20	25	26	5	2	1	36	9	1	0	26	5	2	40
15	3	40	45	52	26	34	17	4	13	20	29	2	5	17	26	0	9	36	2
16	3	37	36	37	29	25	20	13	4	5	8	17	2	2	5	9	0	9	17
17	3	52	45	40	50	34	41	40	13	8	5	50	17	5	2	36	9	0	50
18	3	58	65	74	40	52	29	10	25	34	45	4	13	29	40	2	17	50	0

Matrice de distances - partition finale

		1	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3	3	3
		1	2	3	4	5	6	7	11	12	15	18	9	13	10	8	16	17	14
1	1	0	1	4	2	2	5	20	34	25	40	58	20	29	25	17	37	52	34
2	1	1	0	1	5	1	8	25	41	26	45	65	17	26	20	16	36	45	29
3	1	4	1	0	10	2	13	32	50	29	52	74	16	25	17	17	37	40	26
4	1	2	5	10	0	4	1	10	20	17	26	40	18	25	25	13	29	50	32
5	1	2	1	2	4	0	5	18	32	17	34	52	10	17	13	9	25	34	20
6	1	5	8	13	1	5	0	5	13	10	17	29	13	18	20	8	20	41	25
7	2	20	25	32	10	18	5	0	2	5	4	10	16	17	25	9	13	40	26
11	2	34	41	50	20	32	13	2	0	9	2	4	26	25	37	17	17	50	36
12	2	25	26	29	17	17	10	5	9	0	5	13	5	4	10	2	2	17	9
15	2	40	45	52	26	34	17	4	2	5	0	2	20	17	29	13	9	36	26
18	2	58	65	74	40	52	29	10	4	13	2	0	34	29	45	25	17	50	40
9	3	20	17	16	18	10	13	16	26	5	20	34	0	1	1	1	5	8	2
13	3	29	26	25	25	17	18	17	25	4	17	29	1	0	2	2	2	5	1
10	3	25	20	17	25	13	20	25	37	10	29	45	1	2	0	4	8	5	1
8	3	17	16	17	13	9	8	9	17	2	13	25	1	2	4	0	4	13	5
16	3	37	36	37	29	25	20	13	17	2	9	17	5	2	8	4	0	9	5
17	3	52	45	40	50	34	41	40	50	17	36	50	8	5	5	13	9	0	2
14	3	34	29	26	32	20	25	26	36	9	26	40	2	1	1	5	5	2	0

Matrice des distances

Exemple 1

Fig. 7.2

III-6-2- EXEMPLE 2

Il s'agit d'analyser une population de pièces mécaniques de révolution regroupant des pignons, clabots , fourreaux, pièces diverses. Les plans des pièces ont été codés à l'aide du code OPITZ à 10 digits : 6 variables qualitatives et 4 variables numériques. Sur la production, on a prélevé un échantillon représentatif de 84 pièces.

On recherche une classification de ces pièces en familles.

Sur les variables qualitatives, nous utilisons la distance du khi-2, d_1, tandis que les variables quantitatives sont reliées par la distance Euclidienne d_2.

La distance totale entre deux individus est :

$$d = \alpha d_1 + (1-\alpha) d_2 \text{ avec}$$

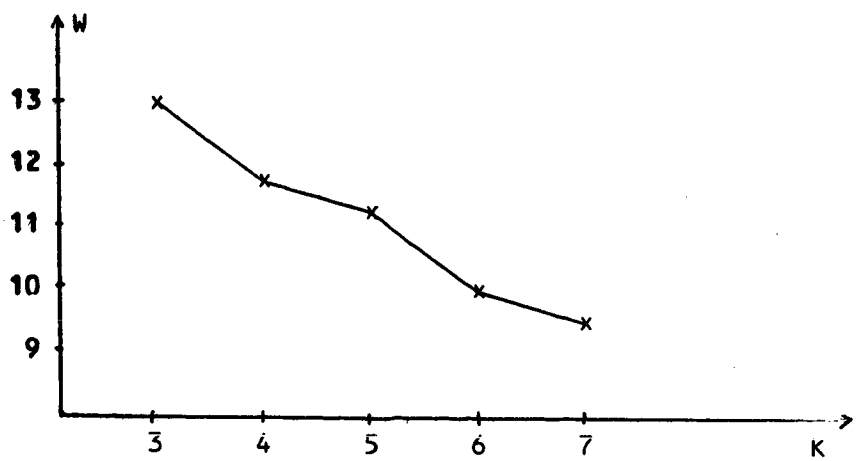
$$\alpha l \, \overline{d}_1 = (1-\alpha)\, t \, \overline{d}_2$$

l = nombre de variables qualitatives

t = nombre de variables quantitatives

$\overline{d}$ = moyenne des distances.

Pour initialiser la procédure de classification, nous effectuons plusieurs tirages aléatoires. Parmi ces tirages nous retenons la partition finale minimisant le critère et ceci pour des nombres différents de classes (fig. 7.3).



Variations de W en fonction du nombre de classes

Fig. 3

La figure 7.3 montre que les partitions en quatre classes et six classes semblent les mieux appropriées.

Les résultats correspondant à ces partitions sont présentés figures 7.4 et 7.5

```
Moyenne  intra-classes=  7.4820107263E+01
Variance intra-classes=  3.3252744730E+03
Moyenne  inter-classes=  1.2644721546E+02
Variance inter-classes=  5.3423961965E+03

Classe  1:  1  2 31  4  5  6  7  8  9 10 11 12 13 14 16 17 18 19 20 22 23 24 25
           26 27 28 29 30  3 32 21 15
Classe  2: 61 62 65 66 68 69 70 71 75 78 82 83 84 56
Classe  3: 67 73 74 76 77 79 80 81 63 64 72
Classe  4: 49 43 41 45 36 37 53 48 35 34 39 44 46 38 47 33 54 59 60 40 51 55 50
           57 58 52 42
```

Meilleure Partition en 4 classes

Fig. 7.4

```
Moyenne  intra-classes=  2.6825219316E+01
Variance intra-classes=  4.0621017597E+02
Moyenne  inter-classes=  5.6198762432E+01
Variance inter-classes=  1.6773556690E+03

Classe  1: 38 45 51 52 43 41 42 44 36 37 49 50 39 58 60 76 77 40
Classe  2: 24 53 54 25 26 27 16 23
Classe  3: 12 13 15 19 20 17 18 21 22  4 11 30
Classe  4: 47 48 56 34 59 84 33 35 46 57 55
Classe  5: 65 66 67 68 69 70 71 72 73 74 75 63 61 78 79 80 81 82 83 62 64
Classe  6: 29 14 31 32 10  2  3  1  5  6  7  8  9 28
```

Meilleure Partition en 6 classes

Fig. 7.5

IV - CLASSEMENT D'UNE NOUVELLE PIECE

IV-2- HEURISTIQUE DE CLASSEMENT

Nous nous intéressons ici à l'aspect décisionnel de l'analyse discriminante. Ainsi, nous supposons connues les classes P_l sur l'ensemble d'apprentissage fini $A \subset E$. Nous recherchons une règle d'affectation F telle que pour un nouvel individu $x \in E$, $F(x)=P_l$ soit la décision d'affecter x à P_l. La décision devant être prise rapidement, nous donnons une règle F simple nécessitant peu de calculs. Une procédure du type "plus proche voisins" est donc rejetée et nous proposons l'heuristique suivante qui ne nécessite que le calcul de K distances, K étant le nombre de classes P_l, $l=1,\ldots,K$.

1. Initialisation. $P=(P,\ldots,P_K)$, $x_i \in A$, card A = n

   1.1. Pour chaque classe P_l, $l=1,\ldots,K$

   $$I_l = \min_i \sum_j m_j \, d(x_i,x_j), \quad \forall x_i,x_j \in P_l$$

   $g_l = x_i$ qui réalise le minimum précédent

   $$M_l = \sum_{i \in P_l} m_i$$

2. Affectation. Soit un nouvel individu $x \in E$

   2.1. Pour chaque classe P_l

   Faire $F_l(x)=M_l \, d(x,g_l) + I_l$

   2.2. x sera affecté à P_r pour

   $$F_r(x) = \min_l F_l(n)$$

3. Fin de procédure.

Cette heuristique revient à rechercher pour chaque classe P_l l'individu g_l le plus proche du centre de gravité de P_l. A g_l est affectée la variance intra classe I_l. Les distances d étant quadratiques, on peut utiliser le théorème de Huygens. Si g_l^* est le centre de gravité de P_l, alors :

$$I_l^* = \sum_i m_i \, d(x_i,g_l^*)$$

d'où

$$D(x,P_l) = M_l \, d(x,g_l^*) + I_l^*$$

n'est autre que la distance, à une constante près, utilisée dans l'algorithme de classification (III-5-2-).

L'erreur introduite par cette heuristique dépend de la distance entre g_l et g_l^*. Pour des classes non creuses, ce qui est notre cas, et compte tenu des propriétés de l'algorithme de classification, cette distance $d(g_l g_l^*)$ est supposée faible par rapport à l'écart-type de la classe P_l.

La même heuristique peut être utilisée dans l'algorithme de classification, ce qui conduit à une simplification des calculs. En annexe, nous présentons cette heuristique de classification et l'appliquons à un exemple de non convergence de la méthode des nuées dynamiques, proposée par Lechevallier (9.).

IV-2- RENFORCEMENT

IV-2-1- APPRENTISSAGE SANS MAITRE

x est affecté à P_l.

Deux cas possibles :

a) Var $(P_l + \{x\})$ décroît. On mémorise simplement x.

b) Var $(P_l + \{x\})$ augmente. On mémorise x et on le comptabilise dans un compteur C.

Lorsque $\frac{C}{n} \geqslant b$ donné , on procède à une mise à jour des classes en effectuant une nouvelle classification.

IV-2-2- APPRENTISSAGE AVEC MAITRE

F(x) affecte x à P_l.

Deux cas sont possibles :

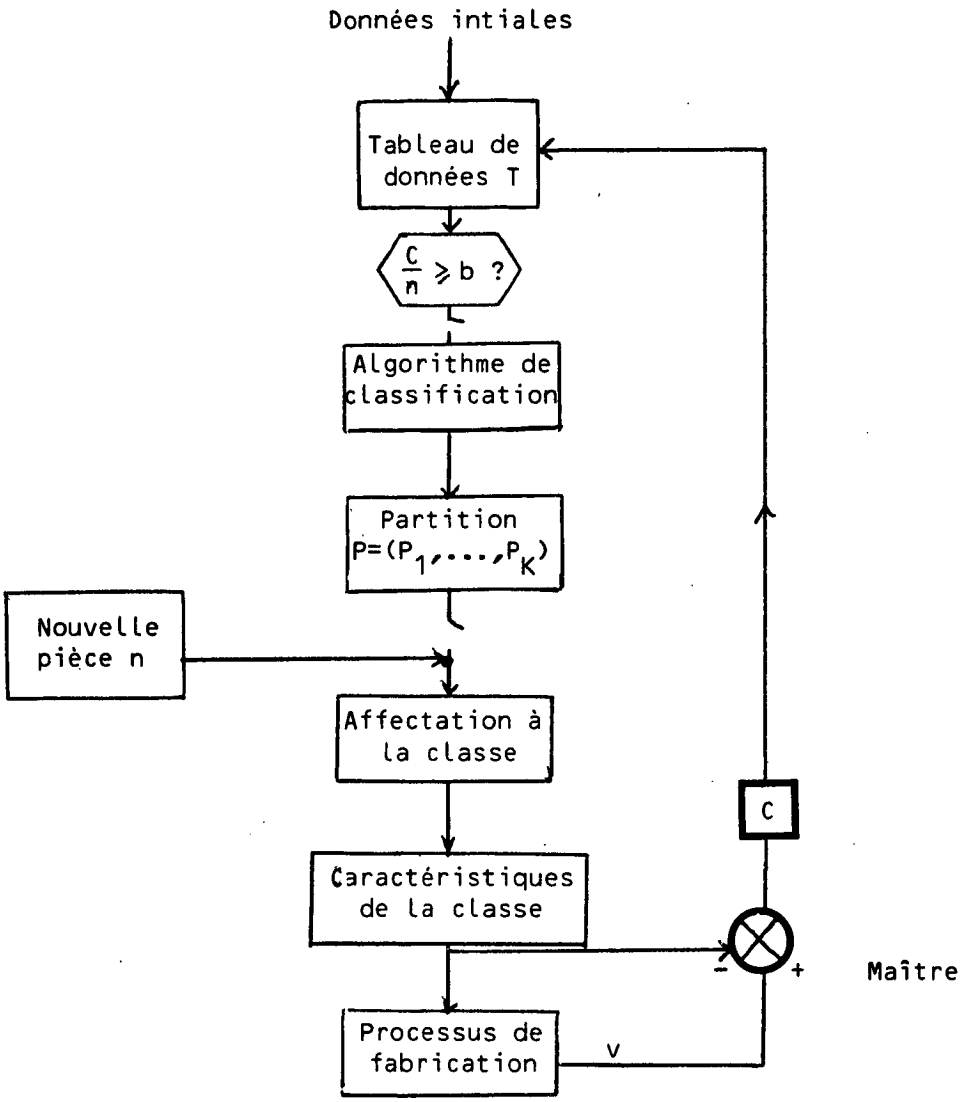
a) l'affectation est bonne et on mémorise simplement x,

b) l'affectation est mauvaise : on mémorise x et on le comptabilise dans C.

Lorsque $\frac{C}{n} \geqslant b$ donné , on procède à une nouvelle classification (fig. 8).

IV-3- CAS DE DISTANCES SUR PLUSIEURS GROUPES

C'est un cas très fréquent en technologie de groupe. Sur un ensemble d'apprentissage A on effectue une classification à partir de la connaissance pour chaque pièce du code morphodimensionnel (1er groupe de variable E_1) et, par exemple, des gammes de fabrication (2ème groupe de variables E_2). Cette classification étant établie, on désire alors pouvoir classer de nouvelles pièces connaissant uniquement le premier groupe de variables. Cette méthode de génération de gammes de fabrication est appelée approche par variante (14.).



Procédure de renforcement

Fig. 8

Le deuxième groupe de variables et considéré alors comme une contrainte que l'on relaxe en diminuant le poids de d_2 après chaque convergence de l'algorithme.

V - CONCLUSION

Dans le cadre des études de la Technologie de Groupe Assistée par Ordinateur, nous avons présenté dans ce papier l'apport de l'analyse des données dans les problèmes de codage , de classification et de classement.

Nous proposons une méthode de classification automatique et de classement qui permet de constituer des familles de produits et de classer des nouveaux produits. Cette méthode tient compte des particularités des codes morphodimensionnels représentant les produits, de la nature différente des variables (qualitatives, quantitatives, ordinales) et de l'appartenance de ces variables à plusieurs groupes d'entités physiques (code morphodimensionnel, gamme de fabrication, devis, etc.).

Une application permet de tester l'efficacité des algorithmes proposés. Ce logiciel est destiné à être intégré à un système plus général de TGAO comprenant en particulier la recherche d'îlots de production et la génération de gammes de fabrication.

REFERENCES

1. J. PEKLENIK, J. GRUM :

"Investigation of the computer aided classification of parts", Annal CIR , vol 29/1/80, pp 319-323.

2. V. EVERSHEIM, F. LOCK :

"Use of multivariable statistical methods for application of group technology in design and process planning department", CIRP, vol 33/1/83, pp 307-311.

3. J. MINOT, Y. LEMOINE, B. MUTEL :

"Implantation Assistée par ordinateur de la Technologie de Groupe", Congrès AFCET Automatique, Besançon, nov 83.

4. B. MUTEL :

"Reconaissance de groupement techonologique par des méthodes d'analyse des données", Congrès International Productique et Robotique, Bordeaux, Mars 84.

5. A. BRUYAND, B. MUTEL :

"Contribution of data analysis method to group technology implementation", Congrès IFIP-IFAC, Prolamat, Paris 83.

6. A. NADIF :

"Transcodage de donnée de production", Rapport no 1 contrat ADEPA-LAEI, juin 84.

7. B. FICHET, B. LE CALVE :

"Structure géométrique des principaux indices de dissimilarités sur signe de présence absence", Statistique et analyse des données, vol 9, no 3, pp 11-44, 1984.

8. J. ESCOFFIER, J. PAGES :

"Comparaison de groupes de variables définies sur le même ensemble d'individus" Rapport de Recherche INRIA no 149, juillet 82.

9. Y. CHEVALIER :

"Optimisation de quelques critères en classification automatique", Thèse de 3ème cycle, Université de Paris VI, juin 74.

10. H. GARCIA, J.M. PROTH :

"Group technology in production management : the short horizon planning level", Rapport de Recherche INRIA no 376.

11. A. NADIF, M. CONSTANTINI, B. MUTEL :

"Mesures de ressemblance de gammes de fabrication", APII, vol 19, no 5, 1985.

12. B. MUTEL, K. MEIER :

"A method for process-planning recognition. Application to Group Technology", Publication proposée à AMS 86, Chicago.

13. J. CHANDON, S. PINSON :

"Analyse typologique : théorie et application", Edition Masson.

14. B. MUTEL :

Cours de F.A.O., DEA de Production Automatisée.

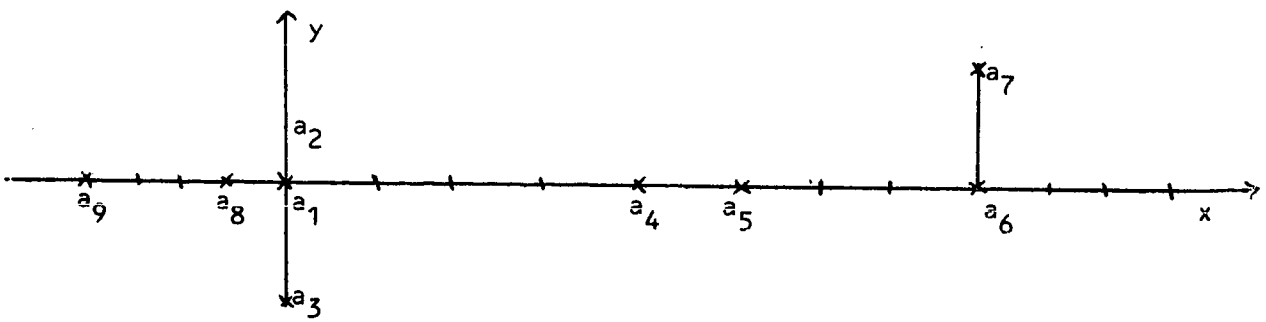
ANNEXE

HEURISTIQUE DE CLASSIFICATION

1. Initialisation
   Partition initiale $P^{(0)} = (P,\ldots,P_K)$, $W^{(0)} = \infty$
2. Pour chaque classe P_l, $l=1,\ldots,K$
   2.1. Pour chaque individu $x_i \in P_l$
      2.1.1. Pour chaque individu $x_j \in P_l$ calculer $I_i = \sum_j m_j\, d(x_j,x_i)$
      $I_l = \min_i I_i$ pour $x_i = g_l$
3. Pour chaque individu x_i, $i=1,\ldots,n$
   3.1. Pour chaque classe P_l, $l=1,\ldots,K$
      Faire $D(x_i,P_l) = d(x_i,g_l) + \frac{1}{M_l} I_l$
      3.1.1. Affecter x_i à P_l si $D(x_i,P_l) = \min_r D(x_i,P_r)$
4. Calculer $W' = \sum_{p=1}^{K} \sum_j m_j \sum_i m_i\, d(x_i,x_j)$
   4.1. Si $W^{(0)} - W' > \varepsilon$ alors $W^{(0)} = W'$ et retourner en 2.
   Sinon, fin de procédure.
5. Fin de procédure.

EXEMPLE

- Les données de cet exemple ainsi que la partition initiale sont tirés de (12).
- La distance entre les individus est la distance Euclidienne.



$P = (P_1, P_2)$

$P_1^{(0)} = (a_1, a_2, a_3 a_4, a_8, a_9)$

$P_2^{(0)} = (a_5, a_6, a_7)$

Tableau T des distances

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9
a_1	0	1	1	16	25	64	68	1	16
a_2	1	0	4	17	26	65	69	$\sqrt{2}$	17
a_3	1	4	0	17	26	65	73	$\sqrt{2}$	17
a_4	16	17	17	0	1	16	20	25	64
a_5	25	26	26	1	0	9	13	36	81
a_6	64	65	65	16	9	0	4	81	144
a_7	68	69	73	20	13	4	0	85	148
a_8	1	$\sqrt{2}$	$\sqrt{2}$	25	36	81	85	0	9
a_9	16	17	17	64	81	144	148	9	0

- ITERATION 1

Pour $P_1^{(0)}$, I est minimum avec $g_1 = a$, et $I_1 = \frac{35}{6}$

Pour $P_2^{(0)}$, I est minimum avec $g_2 = a_6$ et $I_2 = \frac{13}{3}$

$W^{(0)} = 10,16$

Construction de la nouvelle partition

$P_1^{(1)} = (a_1, a_2, a_3, a_8, a_9)$

$P_2^{(0)} = (a_4, a_5, a_6, a_7)$

- ITERATION 2

Pour $P_1^{(1)}$, I est minimum avec $g_1 = a_8$ et $I_1 = \frac{12,8}{5}$

Pour $P_2^{(1)}$, I est minimum avec $g_2 = a_5$ et $I_2 = \frac{23}{4}$

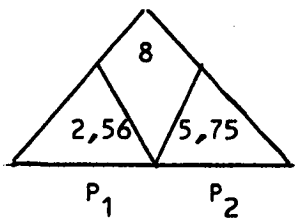
$W^{(1)} = 8,31$

Construction de la nouvelle partition

$P_1^{(2)} = (a_1, a_2, a_3, a_8, a_9)$

$P_2^{(2)} = (a_4, a_5, a_6, a_7)$

$W^{(2)} = 8,31$: convergence



Avec cet exemple, la méthode présentée en (12.) page 23 (noyau constitué de deux individus et d'un centre de gravité) ne convergeait pas.

LISTING DES PROGRAMMES

```
 1  (*$R+,U+ *)
 2  program creemat(input,output);
 3
 4  (* ** programme de creation et de **
 5     ** manipulation des donnees    **
 6
 7      --version du 10/01/86--------
 8
 9      nombre maxi de lignes   = 100
10      nombre maxi de colonnes = 10
11
12  *)
13
14  const nmax=100;
15        mmax=10;
16  type  typeligne=1..nmax;
17        typecol  =1..mmax;
18        bvect    = array[typecol] of boolean;
19        ivect    = array[typecol] of integer;
20        rvect    = array[typecol] of real;
21  var   matb     : array[typeligne] of ivect;
22        matr     : array[typeligne] of rvect;
23        pi       : array[typeligne] of integer;
24        filedim  : file of integer;
25        filemat  : file of bvect;
26        filematr: file of rvect;
27        filepi   : file of integer;
28        rep      : char;
29        m,n      : integer;
30
31  (*$I wait.inc*)
32  (*$I litmn.inc*)
33  (*$I litmat.inc*)
34  (*$I litmr.inc*)
35  (*$I ecritmn.inc*)
36
37  procedure ecritmat;
38  var i:typeligne;
39      j:typecol;
40      t:bvect;
41  begin
42    ecritmn;
43    assign(filemat,'matb.dta');
44    rewrite(filemat);
45    for i:=1 to n do
46    begin
47      for j:=1 to m do
48        t[j]:=(matb[i][j]=1);
49      write(filemat,t)
50    end;
51    close(filemat)
52  end;
53
54  procedure ecritmatrel;
55  var i:typeligne;
```

```
56      j:typecol;
57  begin
58    ecritmn;
59    assign(filematr,'matr.dta');
60    rewrite(filematr);
61    for i:=1 to n do
62      write(filematr,matr[i]);
63    close(filematr)
64  end;
65
66
67  procedure newmatbol;
68  {nouvelle matrice}
69  var i:typeligne;
70      j:typecol;
71
72  begin
73    writeln('Creation matrice booleenne');
74    writeln;
75    write('Nombre d''individus=');
76    readln(n);
77    write('Nombre de parametres=');
78    readln(m);
79    for i:=1 to n do
80    begin
81      writeln('individu ',i:3);
82      writeln('------------');
83      for j:=1 to m do
84      begin
85        write(j:3,':');
86        readln(matb[i][j])
87      end
88    end;
89    ecritmat
90  end; {newmatbol}
91
92  procedure affmatbol;
93  {affiche mat. booleenne}
94  var i:typeligne;
95      j:typecol;
96
97  begin
98    litmn;
99    litmat;
100   clrScr;
101   writeln('          Matrice booleenne');
102   writeln;
103   for i:=1 to n do
104   begin
105     for j:=1 to m do
106       if matb[i][j]=1 then write('1') else write(' ');
107     writeln;
108     if (i=n) or (i mod 20=0)then wait
109   end
110 end; {affmatbol}
```

```
111
112  procedure newmatrel;
113  var i:typeligne;
114      j:typecol;
115  begin
116    writeln('Creation matrice reelle');
117    writeln;
118    write('Nombre d''individus=');
119    readln(n);
120    write('Nombre de parametres=');
121    readln(m);
122    for i:=1 to n do
123    begin
124      writeln('individu ',i:3);
125      writeln('------------');
126      for j:=1 to m do
127      begin
128        write(j:3,':');
129        readln(matr[i][j])
130      end
131    end;
132    ecritmatrel
133  end;
134
135  procedure affmatrel;
136  var i:typeligne;
137      j:typecol;
138
139  begin
140    litmn;
141    litmatrel;
142    clrScr;
143    writeln('              Matrice reelle');
144    writeln;
145    for i:=1 to n do
146    begin
147      for j:=1 to m do
148        write(matr[i][j]:3:1,' ');
149      writeln;
150      if (i=n) or (i mod 20=0) then wait
151    end
152  end;
153
154  procedure transrb;
155  var i:typeligne;
156      j:typecol;
157      u:integer;
158  begin
159    litmn;
160    litmatrel;
161    for i:=1 to n do
162      for j:=1 to m do
163      begin
164        if matr[i][j]<>0 then u:=1 else u:=0;
165        matb[i][j]:=u
```

```
166       end;
167     ecritmat
168   end;
169
170   procedure newponder;
171   var i:typeligne;
172       rep:char;
173   procedure ecritpond;
174   var i:typeligne;
175   begin
176     assign(filepi,'PI.DTA');
177     rewrite(filepi);
178     for i:=1 to n do
179       write(filepi,pi[i]);
180     close(filepi)
181   end;
182
183   begin
184     writeln('          Ponderations');
185     litmn;
186     writeln(' Initialisation de toutes');
187     writeln(' les ponderations a 1 ? (o/n)');
188     read(rep);
189     if rep='o' then
190     begin
191       for i:=1 to n do
192         pi[i]:=1
193     end;
194     ecritpond
195   end;
196
197   procedure litpond;
198   var i:typeligne;
199   begin
200     litmn;
201     assign(filepi,'PI.DTA');
202     reset(filepi);
203     for i:=1 to n do
204       read(filepi,pi[i]);
205     close(filepi)
206   end;
207
208   procedure affponder;
209   var i:typeligne;
210   begin
211     litpond;
212     writeln('  Ponderations:');
213     for i:=1 to n do
214     begin
215       writeln('Individu ',i:3,' P=',pi[i]:3);
216       if (i=n) or (i mod 20=0) then wait
217     end
218   end;
219
220   procedure modif;
```

```
221  var u:integer;
222      i:typeligne;
223      j:typecol;
224  begin
225    writeln('Modification de la matrice reelle:');
226    litmn;
227    litmatrel;
228    repeat
229      write('No de l''individu (ou 0)=');
230      readln(u);
231      if u<>0 then
232      begin
233        i:=u;
234        for j:=1 to m do
235          write(matr[i,j]:5:2);
236        writeln;
237        write('No du parametre (ou 0)=');
238        readln(u);
239        if u<>0 then
240        begin
241          j:=u;
242          writeln('ancienne valeur=',matr[i,j]:5:2);
243          write(  'nouvelle valeur=');readln(matr[i,j])
244        end
245      end
246    until u=0;
247    ecritmatrel
248  end;
249
250  procedure menu(var rep:char);
251
252  begin
253    clrscr;
254    writeln('     Manipulation des fichiers');
255    writeln('     -------------------------');
256    writeln;
257    writeln('  [1] Creation matrice booleenne');
258    writeln('  [2] Affichage matrice booleenne');
259    writeln('  [3] Creation matrice reelle');
260    writeln('  [4] Affichage matrice reelle');
261    writeln('  [5] Transformation reelle->booleenne');
262    writeln('  [6] Creation des ponderations');
263    writeln('  [7] Affichage des ponderations');
264    writeln('  [8] Modifications');
265    writeln('  [9] Fin');
266    write  ('  -->');
267    readln(rep)
268  end; {end}
269
270
271  begin {creemat}
272    repeat
273      menu(rep);
274      case rep of
275        '1':newmatbol;
```

```
276         '2':affmatbol;
277         '3':newmatrel;
278         '4':affmatrel;
279         '5':transrb;
280         '6':newponder;
281         '7':affponder;
282         '8':modif;
283         '9':begin end
284       end (case)
285     until (rep='9')
286   end.
```

```
 1  Program Makedis(input,output);
 2  (*$U+,R+,C+ *)
 3
 4  (* Ce programme necessite que les fichiers 'DIM.DTA' et 'MATR.DTA'
 5     aient ete crees par le programme de chargement 'Creemat'
 6
 7
 8   * Creation du tableau de distances du Khi-2 sur les variables qualitatives
 9     Les valeurs des modalites doivent etre entieres.
10   * Creation du tableau des distances Euclidiennes sur les variables
11     quantitatives.
12   * Le tableau final est place dans le fichier filedis (DIS.DTA) .
13
14     Version du 15/01/86 *)
15
16  const nmax=100;              { nb maxi d'individus }
17        mmax=10;               { nb maxi de variables }
18        pmax=4950;             { nmax(nmax+1)/2 }
19        pqmax=7;               { nb maxi de variables qualitatives }
20        modalmax=9;            { 0..9 modalites/variable }
21        TotModal=69;           { nb total de modalites=(modalmax+1)*pqmax-1 }
22        epsi=1E-9;
23  type  typeligne=1..nmax;
24        typecol  =1..mmax;
25        typeQ    =1..pqmax;
26        rvect    = array[typecol] of real;
27        disvect  = array[typeligne] of real;
28  var   matr     : array[typeligne] of rvect;
29        filedim  : file of integer;
30        filematr: file of rvect;
31        filedis  : file of disvect;
32        rep      : char;
33        m,n      : integer;
34
35        lambda   : integer;
36        pq       : typeQ;
37        qual     : array[typecol] of boolean;
38        z        : array[typeligne,typeQ] of 0..modalMax;
39        tabToe   : array[typecol] of real;
40
41        Kmoy, Tok, K1, Emoy, Toe, E1, pq2 :real;
42
43  procedure litmn;
44  begin
45    assign(filedim,'DIM.DTA');
46    reset(filedim);
47    read(filedim,m);
48    read(filedim,n);
49    close(filedim)
50  end;
51
52  procedure litmatrel;
53  var i:typeligne;
54      j:typecol;
55  begin
```

```
  litmn;
  assign(filematr,'matr.dta');
  reset(filematr);
  for i:=1 to n do
    read(filematr,matr[i]);
  close(filematr)
end;

Procedure EntreeDonnees;
var j   :typeQ;
    j1  :typecol;
begin
  ClrScr;
  writeln('-------------------------------------');
  writeln('!                                   !');
  writeln('! CREATION DU FICHIER DES DISTANCES !');
  writeln('!                                   !');
  writeln('-------------------------------------');
  writeln;
  write('Nombre de variables qualitatives=');
  readln(pq);
  pq2:=sqr(pq);
  for j1:=1 to m do
    qual[j1]:=false;
  writeln('Donnez le rang des variables qualitatives:');
  for j:=1 to pq do
  begin
    write(j:3,':');
    readln(j1);
    qual[j1]:=true
  end;
  write('Nombre de modalites par variable (constant)=');
  readln(lambda)

end;


Procedure CreeFileDis;

type typeModal=0..TotModal;
var rep     : char;
    tabMod  : array[typemodal] of 0..Modalmax;
    (*tabIndex est cree pour accelerer les acces*)
    tabIndex: array[typemodal] of typeQ;
    m_j     : array[typeModal] of real;
    s       : typeModal;
    alpha   : real;

procedure CreeTabIndex;
var j : typeModal;
begin
  { tableau d'index q=f(j) }
  writeln('Tableau d''index:');
  for  j:=0 to s do
  begin
```

```
111       tabIndex[j]:=trunc(j/lambda)+1;
112       write(tabIndex[j]:2)
113     end;
114     writeln
115   end;
116
117   procedure CreeTabMod;
118   { tableau des modalites tabMod }
119   var j : typeModal;
120   begin
121     writeln('Tableau des modalites:');
122     for j:=0 to s do
123     begin
124       tabMod[j]:=j mod lambda;
125       write(tabMod[j]:2)
126     end;
127     writeln
128   end;
129
130   procedure Makez;
131   { creation du tableau z }
132   var q : integer;
133       j : typecol;
134       i : typeligne;
135   begin
136     writeln('Z: Tableau des Variables qualitatives:');
137     for i:=1 to n do
138     begin
139       write(i:2,':');
140       q:=0;
141       for j:=1 to m do
142        if qual[j] then
143         begin
144           q:=q+1;
145           z[i,q]:=trunc(matr[i,j]);
146           write(z[i,q]:2)
147         end;
148       writeln
149     end
150   end;
151
152   procedure CreeM_j;
153   var j   : typemodal;
154       som : real;
155       q   : typeQ;
156       i   : typeligne;
157       l1  : 0..modalMax;
158   begin
159     writeln('Calcul des m.j ');
160     for j:=0 to s do
161     begin
162       som:=0;
163       for i:=1 to n do
164       begin
165         q:=tabIndex[j];
```

```
166         if z[i,q]=tabMod[j] then som:=som+1
167       end;
168       m_j[j]:=som
169     end;
170     writeln;
171     { edition des m.j }
172     write('     ');
173     for l1:=0 to lambda-1 do
174       write(l1:4);
175     writeln;
176     for q:=1 to pq do
177     begin
178       write(q:2,':');
179       for l1:=0 to lambda-1 do
180         write(m_j[lambda*(q-1)+l1]:4:0);
181       writeln
182     end;
183     writeln;
184     for j:=0 to s do if m_j[j]<>0 then
185       m_j[j]:=1/m_j[j]
186   end;
187
188   procedure calc_m_j;
189
190   begin                              { calc_m_j }
191     s:=lambda*pq-1;
192     writeln('Nombre total de modalites=',s:3);
193     CreeTabIndex;
194     CreeTabMod;
195     Makez;
196     CreeM_j
197   end;
198
199   function Khi2(u,v:typeligne):real;
200                                      { distance du khi2 entre u et v =d^2(u,v) }
201   var q:typeQ;
202       j:typeModal;
203       som:real;
204   begin                              { creeFileKhi2 }
205     som:=0;
206     for j:=0 to s do
207     begin
208       q:=tabIndex[j];
209       if m_j[j]<>0 then
210         if z[u,q]<>z[v,q] then som:=som+m_j[j]
211     end;
212     Khi2:=som/pq2
213   end; { Khi2 }
214
215   Procedure K_MoyEcart;
216   { moyenne et ecart type de K=d(u,v) }
217   var u,v : typeligne;
218       tot : integer;
219       sig,sig2,x:real;
220       p   : integer;
```

```
221  begin
222    writeln('          Distances du Khi-2:');
223    writeln('Calcul de la moyenne et de l''ecart type.');
224    sig:=0;
225    sig2:=0;
226    p:=trunc(n*(n+1)/2);
227    for u:=1 to n do
228      for v:=u to n do
229      begin
230        x:=Khi2(u,v);
231        sig:=sig+x;
232        sig2:=sig2+sqr(x)
233      end;
234      tot:=trunc(n*(n+1)/2);
235      Kmoy:=sig/tot;
236      Tok:=sqrt(sig2/tot-sqr(Kmoy));
237      writeln;
238      writeln('Moyenne=',Kmoy:8:3,',Ecart Type=',Tok:8:3)
239  end;
240
241  procedure CalcMTo;
242  { calcule les moyennes et ecart type des variables quantitatives }
243  var j   : typecol;
244      i   : typeligne;
245      Moy : array[typecol] of real;
246      s,s2:real;
247  begin
248    writeln('Calcul des moyennes et ecarts types des variables quantitatives.');
249    write('Variable :');
250    for j:=1 to m do if (not qual[j]) then
251    begin
252      write(j:5);
253      s:=0;
254      s2:=0;
255      for i:=1 to n do
256      begin
257        s:=s+matr[i,j];
258        s2:=s2+sqr(matr[i,j])
259      end;
260      Moy[j]:=s/n;
261      TabToe[j]:=sqrt(s2/n-sqr(s/n))
262    end;
263    writeln;
264    write('Moyenne  :');
265    for j:=1 to m do if (not qual[j]) then
266      write(Moy[j]:5:2);
267    writeln;
268    write('Ecart Typ:');
269    for j:=1 to m do if (not qual[j]) then
270      write(TabToe[j]:5:2);
271    writeln;
272    writeln
273  end; {CalcMTo}
274
275  Function Euclide(u,v:typeligne):real;
```

```
276  { distance euclidienne au carre entre les variables quantitatives
277    des individus u et v sur les variables normees }
278  var x:real;
279      j:typecol;
280  begin
281    x:=0;
282    for j:=1 to m do if (not qual[j]) then
283      x:=x+sqr(matr[u,j]-matr[v,j])/TabToe[j];
284    Euclide:=x
285  end; { Euclide }
286
287
288  Procedure E_MoyEcart;
289  { moyenne et ecart type des distances Euclidiennes }
290  var u,v : typeligne;
291      tot : integer;
292      p   : 1..pmax;
293      sig,sig2,x:real;
294  begin
295    writeln('         Distances Euclidiennes:');
296    writeln('Calcul de la moyenne et de l''ecart type.');
297    sig:=0;
298    sig2:=0;
299    p:=trunc(n*(n+1)/2);
300    for u:=1 to n do
301      for v:=u to n do
302      begin
303        x:=Euclide(u,v);
304        sig:=sig+x;
305        sig2:=sig2+sqr(x)
306      end;
307      tot:=trunc(n*(n+1)/2);
308      Emoy:=sig/tot;
309      Toe:=sqrt(sig2/tot-sqr(Emoy));
310      writeln;
311      writeln('Moyenne=',Emoy:8:3,',Ecart Type=',Toe:8:3)
312  end;
313
314  Procedure Creefic;           { procedure de creation du fichier des distances }
315  var u,v   : typeligne;
316      p     : 0..pmax;
317      x,beta: real;
318      vect  : disvect;
319  begin
320    ClrScr;
321    p:=sqr(n);
322    writeln('Creation du fichier des distances Khi2,Euclidienne..,');
323    writeln('(',p:6,' valeurs calculees.)');
324    assign(Filedis,'DIS.DTA');
325    rewrite(Filedis);
326    beta:=1-alpha;
327    for u:=1 to n do
328    begin
329      for v:=1 to n do
330        vect[v]:=alpha*Khi2(u,v) + beta*Euclide(u,v);
```

```
331       write(Filedis,vect);
332     end;
333     close(Filedis);
334     writeln;
335     writeln('Termine.')
336   end; {Creefic}
337
338   begin                          { CreeFileDis }
339     Calc_m_j;
340     write('Calcul des moy. et ecart type des Khi-2 o/n ?');
341     readln(rep);
342     if rep ='o' then
343       K_MoyEcart;
344     CalcMTo;
345     E_MoyEcart;
346     writeln('Valeur du coefficient alpha :');
347     writeln('    ->0 pondere les variables Quantitatives.');
348     writeln('    ->1    "     "      "     Qualitatives.');
349     write  ('     =');readln(Alpha);
350     Creefic
351   end;
352
353   begin                          { PROGRAMME PRINCIPAL }
354     EntreeDonnees;
355     litmn;
356     litmatrel;
357     CreeFileDis
358   end.
```

```
 1  program ddc(input,output);
 2  { le 13/01/86 }
 3                          {-------------------------------------}
 4                          {CLASSIFICATION DIRECTE D'UN ENSEMBLE }
 5                          {   D'INDIVIDUS EN k CLASSES SUR LA   }
 6                          {     MATRICE DES DISTANCES.          }
 7                          {-------------------------------------}
 8
 9  (* ce programme necessite les fichiers DIM.DTA,DIS.DTA,
10     le fichier de resultats s'appelle RESULT.TXT *)
11
12  (*$R+,U+,C+ *)
13  const nmax=100;
14        mmax=10;
15        kmax=7;
16        col=80;
17  type  typeligne =1..nmax;
18        typecol   =1..mmax;
19        typeclasse=1..kmax;
20        typevect  =array[typeligne] of real;
21  var   pos      : array[typeligne] of typeligne;
22        af       : array[typeligne] of typeclasse;
23        filedim  : file of integer;
24        filedis  : file of typevect;
25        fileres  : text;
26        buf      : array[typeligne] of real;
27        rep      : char;
28        m,n,init: integer;
29        k        : typeclasse;
30        print    : boolean;
31        trait    : string[col];
32  (*$I wait.inc*)
33  (*$I litmn.inc*)
34
35
36  procedure litBuf(i:typeligne);
37  begin
38    seek(filedis,i-1);
39    read(filedis,Buf)
40  end;
41
42  procedure initpos;
43  var i:typeligne;
44  begin
45    for i:=1 to n do
46      pos[i]:=i
47  end;
48
49  procedure ResetPart;
50  { Remet a 1 la partitition }
51  var i:typeligne;
52  begin
53    for i:=1 to n do
54      af[i]:=1
55  end;
```

```
 56
 57
 58 procedure afficheDis;
 59 { Edite la matrice des distances sur imprimante ou ecran }
 60 var i,i1:typeligne;
 61     rep :char;
 62
 63
 64
 65 begin
 66   writeln('Matrice des distances, impression:');
 67   if print then write('f)ichier ');
 68   writeln('e)cran,n)on ');
 69   repeat
 70     write('-->');readln(rep);
 71   until (rep in ['f','e','n']);
 72   if ((not print) and (rep='f')) then rep:='n';
 73   case rep of
 74   'e':begin
 75         writeln;
 76         writeln('     Matrice de distances   ');
 77         writeln;
 78         write('       ');
 79         for i:=1 to n do
 80           write(af[i]:2,'    ');
 81         writeln;
 82         write('       ');
 83         for i:=1 to n do
 84           write(pos[i]:2,'    ');
 85         writeln;
 86         for i:=1 to n do
 87         begin
 88           litbuf(pos[i]);
 89           write(pos[i]:2,' ',af[i]:2,' ');
 90           for i1:=1 to n do
 91             write(buf[pos[i1]]:4:2,' ');
 92           writeln
 93         end
 94       end;
 95   'f':begin
 96         writeln(fileres,'     Matrice de distances   ');
 97         writeln(fileres);
 98         write(fileres,'       ');
 99         for i:=1 to n do
100           write(fileres,af[i]:2,'    ');
101         writeln(fileres);
102         write(fileres,'       ');
103         for i:=1 to n do
104           write(fileres,pos[i]:2,'    ');
105         writeln(fileres);
106         for i:=1 to n do
107         begin
108           litbuf(pos[i]);
109           write(fileres,pos[i]:2,' ',af[i]:2,' ');
110           for i1:=1 to n do
```

```
111              write(fileres,buf[pos[i1]]:5:2,' ');
112            writeln(fileres)
113          end
114        end;
115      'n':begin end
116    end (*case*)
117  end;
118
119
120  procedure tri;
121  { Trie les positions des individus ( pos[] ) selon
122    la classification ( af[] )   }
123
124  var i,i1,j: typeligne;
125      ik    : typeclasse;
126  begin
127    for i:=1 to n-1 do
128      for i1:=i+1 to n do
129        if af[i]>af[i1] then
130        begin
131          j:=pos[i];pos[i]:=pos[i1];pos[i1]:=j;
132          ik:=af[i];af[i]:=af[i1];af[i1]:=ik
133        end
134  end;
135
136  (**************************
137   *                        *
138   * PROCEDURES PRINCIPALES *
139   *                        *
140   **************************)
141
142  procedure ddc_Proc(k:typeclasse;init:integer);
143  const maxit=10;
144        epsi =0.01;
145
146  var   critab : array[0..maxit] of real;
147        t      : 0..maxit;
148        cardk  : array[typeclasse] of integer;
149
150  procedure initPart(methode:integer);
151  (* Generation de la partition initiale *)
152
153  var i,i1       : typeligne;
154      ik         : typeclasse;
155      ok         : array[typeclasse] of boolean;
156      okall      : boolean;
157
158  begin
159    case methode of
160    1:(*generation aleatoire*)
161    begin
162    (*note: generateur aleatoire pseudo uniforme
163              'randomize' initialise le generateur,
164              'random' genere un nombre n appartenant a [0,1[
165    *)
```

```
166       randomize;
167       repeat
168         for ik:=1 to k do
169           ok[ik]:=false;
170         for i:=1 to n do
171         begin
172           ik:=trunc(random*k)+1;
173           ok[ik]:=true;
174           af[i]:=ik
175         end;
176         okall:=true;
177         for ik:=1 to k do
178           okall:=okall and ok[ik]
179       until okall
180     end;
181     2:(*generation manuelle*)
182     begin
183       writeln('Donnez les affectations des');
184       writeln(n:2,' lignes aux ',k:2,' classes');
185       for i:=1 to n do
186       begin
187         write(i:2,':');
188         readln(af[i])
189       end
190     end
191     end; (*case*)
192
193     for ik:=1 to k do
194       cardk[ik]:=0;
195     for i:=1 to n do
196       cardk[af[i]]:=cardk[af[i]]+1
197   end;
198
199   procedure new_clustering(var w:real);
200   (* Definition de la nouvelle partition *)
201   var i       : typeligne;
202       af1     : array[typeligne] of typeclasse;
203       ik      : typeclasse;
204       inertie : real;
205       cardk1  : array[typeclasse] of integer;
206
207   function inert(k:typeclasse; buf:typevect):real;
208   (* 'Inertie' de la classe k par rapport a l'individu de rang i *)
209   var j:typeligne;
210       x:real;
211   begin
212     x:=0;
213     for j:=1 to n do
214       if af[j]=k then
215         x:=x+buf[pos[j]];
216     inert:=x
217   end;
218
219   procedure BestClasse(var bestk : typeclasse;var inertie : real);
220   (* Meilleure Classe pour l'individu de rang i selon le critere
```

```
221     1/card[ik]*inert(ik,i)          *)
222
223  var min,x    : real;
224      ik,index : typeclasse;
225      j:typeligne;
226  begin
227    min:=1E36;
228    for ik:=1 to k do
229      if cardk[ik]<>0 then
230      begin
231          x:=1/cardk[ik]*inert(ik,buf);
232          if x<min then
233          begin
234            min:=x;
235            index:=ik
236          end
237      end;
238    bestk:=index;
239    inertie:=min
240  end;
241
242  begin (* new_clustering *)
243    for ik:=1 to k do
244      cardk1[ik]:=0;
245    w:=0;
246    for i:=1 to n do
247    begin
248      litbuf(pos[i]);
249      bestClasse(af1[i],inertie);
250      w:=w+inertie;
251      cardk1[af1[i]]:=cardk1[af1[i]]+1
252    end;
253
254    for i:=1 to n do
255      af[i]:=af1[i];
256    for ik:=1 to k do
257      cardk[ik]:=cardk1[ik]
258  end; (* new_clustering *)
259
260  procedure afficheClasses;
261  var i : typeligne;
262      ik: typeclasse;
263      c :integer;
264  begin
265    for ik:=1 to k do
266    begin
267      if print then
268      begin
269        write(fileres,'Classe',ik:3,':');
270        c:=10
271      end;
272      write(    'Classe',ik:3,':');
273      for i:=1 to n do
274        if af[i]=ik then
275        begin
```

```
276           if print then
277           begin
278              write(fileres,pos[i]:3);
279              c:=c+3;
280              if c>col-3 then
281                        begin
282                          writeln(fileres);
283                          write(fileres,'          ');
284                          c:=10
285                        end
286           end;
287           write(pos[i]:3)
288        end;
289        if print then writeln(fileres);
290        writeln
291    end;
292    writeln
293  end;
294
295  procedure TailleClasses;
296  { Cumul des distances par bloc, moyennes variances intra-classes }
297  { Edition format !_0000.0_! }
298  const maxtrait=91;  { 9*kmax+1 }
299  var tab : array [typeclasse,typeclasse] of real;
300      u,v : typeclasse;
301      i,j : typeligne;
302      s,s2,moyIntra,moyInter,VarIntra,VarInter : real;
303      nb,c : integer;
304      trait,trait2 : string[maxtrait];
305  begin
306    { Sommes }
307    for u:=1 to k do
308      for v:=1 to k do
309        tab[u,v]:=0;
310    for i:=1 to n do
311    begin
312      litbuf(pos[i]);
313      for j:=1 to n do
314        tab[af[i],af[j]]:=tab[af[i],af[j]]+buf[pos[j]]
315    end;
316
317    { Moyenne, Variance intra-classes }
318    s:=0;
319    s2:=0;
320    nb:=0;
321    for u:=1 to k do
322    begin
323      s:=s+tab[u,u];
324      s2:=s2+sqr(tab[u,u]);
325      nb:=nb+1
326    end;
327    moyIntra:=s/nb;
328    VarIntra:=s2/nb-sqr(MoyIntra);
329    { Variance inter-classes non vides }
330    s:=0;
```

```
331     s2:=0;
332     for u:=1 to k do
333       for v:=1 to k do
334       begin
335         s:=s+tab[u,v];
336         s2:=s2+sqr(tab[u,v])
337       end;
338     nb:=sqr(k);
339     MoyInter:=s/nb;
340     VarInter:=s2/nb-sqr(MoyInter);
341
342     { Edition }
343     writeln('  Moyenne  intra-classes=',MoyIntra);
344     writeln('  Variance intra-classes=',VarIntra);
345     writeln('  Moyenne  inter-classes=',MoyInter);
346     writeln('  Variance inter-classes=',VarInter);
347
348     if print then
349     begin
350       trait:='-';
351       trait2:='!';
352       for c:=1 to 9*k do
353         trait:=trait+'-';
354       for u:=1 to k do
355         trait2:=trait2+'        !';
356       writeln(fileres);
357       writeln(fileres,'          Cumul des distances par bloc');
358       writeln(fileres,'          ----------------------------');
359       writeln(fileres);
360       writeln(fileres,trait);
361       for u:=1 to k do
362       begin
363         writeln(fileres,trait2);
364         write(fileres,'!');
365         for v:=1 to k do
366           write(fileres,' ',tab[u,v]:6:1,' !');
367         writeln(fileres);
368         writeln(fileres,trait2);
369         writeln(fileres,trait)
370       end;
371       writeln(fileres);
372       writeln(fileres,'  Moyenne  intra-classes=',MoyIntra);
373       writeln(fileres,'  Variance intra-classes=',VarIntra);
374       writeln(fileres,'  Moyenne  inter-classes=',MoyInter);
375       writeln(fileres,'  Variance inter-classes=',VarInter);
376       writeln(fileres)
377     end
378   end;   { TailleClasses}
379
380   begin (*ddc_proc *)
381     initPart(init);
382     writeln('Partition initiale:');
383     if print then writeln(fileres,'Partiton initiale');
384     afficheClasses;
385     if print then writeln(fileres,trait);
```

```
386    t:=0;
387    critab[t]:=0;
388
389  (*---------debut du processus iteratif---------*)
390    repeat
391      t:=t+1;
392      if print then writeln(fileres);
393      writeln(    'Iteration ',t:3,':');
394      new_clustering(critab[t]);
395      tri;
396      afficheClasses;
397      if print then writeln(fileres,'A l''iteration ',t:3,': Critere=',critab[t]);
398      writeln(    'Critere=',critab[t]);
399      if print then writeln(fileres,trait);
400      writeln;
401    until (abs(critab[t]-critab[t-1])<epsi)
402       or (t=maxit);
403    if t=maxit then
404    begin
405      writeln('  ** Nombre maxi d''iterations atteint');
406      writeln;
407      if print then
408               begin
409                 writeln(fileres,'  ** Nombre maxi d''iterations atteint');
410                 writeln(fileres)
411               end
412    end;
413    TailleClasses;
414    AfficheClasses
415  (*---------  fin du Processus iteratif---------*)
416
417  end; {ddc_proc}
418
419  procedure EntreeDonnees(var k : typeclasse;var m : integer;var print :boolean);
420  var rep:char;
421      c:1..col;
422  begin
423    ClrScr;
424    writeln('*******************************************');
425    writeln('*                                         *');
426    writeln('* programme ddc : Classification directe  *');
427    writeln('*       sur matrice des distances         *');
428    writeln('*                                         *');
429    writeln('*******************************************');
430    writeln;
431    litmn;
432    initpos;
433    resetPart;
434    write('Resultats intermediaires sur fichier (o/n) ?');
435    readln(rep);
436    if rep='o' then
437               begin
438                 assign(Fileres,'Result.txt');
439                 rewrite(Fileres);
440                 writeln(Fileres,'    Classification directe sur la matrice des distan
```

```
441                writeln(Fileres,'
442                writeln(Fileres);
443                print:=true
444              end
445              else print:=false;
446    writeln;
447    affichedis;
448    write('Nombre de classes=');
449    readln(k);
450    writeln('Partition initiale:');
451    write  ('a)leatoire m)anuelle ?');
452    readln(rep);
453    if rep='m' then m:=2
454               else m:=1;
455    writeln;
456
457    trait:='-';
458    for c:=1 to col do
459      trait:=trait+'-'
460  end;
461
462  begin (*principal = DDC *)
463    Assign(filedis,'DIS.DTA');
464    reset(filedis);
465    EntreeDonnees(k,init,print);
466    ddc_proc(k,init);
467
468    close(Filedis);
469    if print then close(Fileres)
470
471  end.
```

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique